DOCUMENT RESUME

ED 183 197 IR 008 139

AUTHOR Dennis, J. Richard
TITLE Practicum Activities for Training Teachers to Use Computers. The Illinois Series on Educational Application of Computers, No. 21e.
INSTITUTION Illinois Univ., Urbana. Dept. of Secondary Education.
SPONS AGENCY EXXON Education Foundation, New York, N.Y.
PUB DATE 79
NOTE 16p.; For related documents, see IR 008 123-139.

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS *Computer Assisted Instruction; *Computer Managed Instruction; Computers; *Educational Objectives; Guidelines; *Learning Activities; *Programing; Task Analysis; Teachers
IDENTIFIERS Computer Education

ABSTRACT

This paper lists 25 objectives which computer-using teachers have designated as important for job success, and presents 11 relevant tasks to serve as a basis for practice and assessment of the objectives. These tasks deal with instructional planning, question episodes, lesson evaluation, tutorial computer assisted instruction, drill and practice, individualization and computer managed instruction, simulation, instructional games, computerized testing, hardware operation, and hardware purchasing. A background reference is cited for each task. (CMV)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED183197

Number 21e

The Illinois Series on Educational Application of Computers

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

PRACTICUM ACTIVITIES FOR

TRAINING TEACHERS TO USE COMPUTERS

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY

J. Richard Dennis

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

College of Education
Department of Secondary Education
University of Illinois
Urbana, Illinois

This work was supported by a grant from
The EXXON Education Foundation

Copyright, 1979, Department of Secondary Education
University of Illinois

IR008139

To the reader,

This is one of a series of documents prepared as resources for the in-service and preservice training of teachers. The general theme of all items in this series is "teaching with or about computers." The booklets are the collective work of a variety of staff and students (both present and former) of the College of Education, University of Illinois at Urbana-Champaign. The project generously was supported between 1977 and 1979 by a grant from the EXXON Education Foundation. All responsibility for the content of these documents rests with the authors and does not reflect necessarily the views of the EXXON Foundation. It was my pleasure to direct this effort throughout.

J. Richard Dennis, PhD
Department of Secondary Education
University of Illinois

Many teacher education programs now include opportunities for a variety of field experiences throughout the undergraduate training program. Teachers intending to utilize a computer system in their teaching will require additional experience to insure an entry level competence with this instructional medium.

An earlier paper in this series (see ISEAC 1e) presented a list of general proficiencies which practicing, computer-using teachers have designated as important for on-job success. We review these general objectives here:

1. Familiarity with computerized teaching materials (i.e., instructional programs) in a variety of fields.
2. Ability to integrate computerized teaching materials into a course.
3. General knowledge of the functioning of CMI (computer-managed-instruction) systems.
4. Understanding of effective design of drill and practice materials.
5. Ability to apply computerized drill and practice in a variety of teaching situations.
6. Familiarity with computer simulations and models.
7. Experience in preliminary design and construction of a simulation.
8. Knowledge of the uses of simulations as teaching tools.
9. Ability to evaluate the effectiveness of a course that uses computerized teaching materials.
10. Ability to determine the computer needs of a school.
11. Ability to draft computer needs and desires of a school to invite proposals/bids from potential suppliers.
12. Ability to be highly critical of suppliers' proposals and their machines.
13. Ability to assemble data about proposed equipment to facilitate decision-making (costs, performance data, hardware characteristics, software support, etc.)
14. Familiarity with instructional games.
15. Knowledge of how to use instructional games appropriately and effectively in teaching.
16. Physical familiarity with computer equipment, i.e., everyday operation and use of a range of different machines.
17. Knowledge of trouble-shooting procedures and means of access to professional help, i.e., knowing how to determine if a piece of equipment is ailing, and if it is, knowing whom to call to fix it.
18. Knowledge of sources for computer materials.
19. Knowledge of how to improve less than adequate instructional computer programs.
20. Ability to evaluate the effectiveness of instructional computer programs.
21. Ability to instruct others in the social role and impact of computers in society.
22. Knowledge of alternative uses of computers in schools e.g., as class-record-keepers, term-paper-editors, etc.
23. Awareness of the value of involving students in the development of computerized instructional materials.

24. Knowledge of processes of involving students in instructional materials development.

25. Computer programming expertise.

The tasks which follow serve as a basis for practice or assessment of these objectives. They also serve to translate the general competencies of a computer using teacher into hands-on activities. The material is organized on a topical basis, each topic being an important aspect of instructional computing. Performance partly is determined by the degree to which execution of each task reflects application of principles or guidelines presented in earlier papers of this series.

These tasks are intended to reflect conditions as they are likely to be found in schools. Clearly, not all schools are alike. Not all tasks will apply to any one school nor to a particular instructional computing system. But the collection should suggest the experiences needed to prepare a teacher to function and to benefit from any instructional computing context they might enter.

Tasks on Instruction Planning

(Background for these tasks may be found in ISEAC #3e.)

1. Prepare a lesson plan for the following six week course. Use the techniques of formal planning and produce at least the first two flowchart drafts. The class meets five days a week for one hour a day. The topic is simple programming in the BASIC language. Details of the content required are given below. Following is a description of the facilities available to you, the constraints placed upon you, and the informal description by the previous teacher of how the class was taught last time it was offered.

The general goal of the course is that students be able to code and debug simple BASIC programs on paper. The students have no computer science background, and have had mathematics through high school algebra. The course is offered through the mathematics department, which has a schoolwide rule that all mathematics courses grade according to the following formula: 50% of the grade is the final exam, and the rest of the grade is five quizzes worth a total of 50% (but not necessarily 10% each.) As this course is an elective, your principal feels that despite the objective grading formula, all students are expected to get A, B, or C. A is defined as 90% to 100%; B as 80% to 89%; C as 70% to 79%; D as 60% to 69%; and failing as below 60%.

Finally, the teachers in the mathematics department are interested in the effectiveness of some available computerized materials. Though you may use any or none as you wish, it has been requested that those components used be evaluated and data supplied to the department on the effectiveness of these components. This data should of course NOT be gathered from student behavior on the computerized components themselves.

The teacher who previously taught the course left a brief description (perhaps not entirely accurate) of what was done during the six weeks. This teacher admitted to having some problems and students did not do as well on the final exam as desired. It also was reported that students did well on the quizzes from week 2 and week 6, but poorly on the quizzes from week 3. Furthermore, students did well on the in-class part of the final, but very poorly on the take-home part.

Here is the informal lesson plan provided by the previous teacher.

> Week 1: Overview and organization - In class, explained what programming is all about, what computers are used for, an overview of BASIC, and the content of the rest of the course.
>
> Week 2: Rules concerning numbers and variables - In class, syntax rules were described and practiced and computerized Quizzes 2 and 3 were taken.
>
> Week 3: Commands and their syntax, semantics - In class, the main commands (assignments, GOTO, IF THEN, PRINT, and END) were explained in terms of proper syntax and uses. Out of class, students were expected to use computerized Tutorials 1, 2, and 3; Drill 1; and Quiz 1.
>
> Week 4: Flowcharting - The week primarily was spent on the theory of algorithms, problem solving, and flowcharting algorithms. Class time was used

to teach the symbols and their purposes in flowcharting. Out of class, students were given a variety of problems (about four) to be flowcharted.

Week 5: Programming, translating flowcharts into statements - Class time was used to show how the flowcharts produced in the previous week could be translated into program statements. As an out of class assignment, three new flowcharts were given which the students were to translate into programs.

Week 6: Debugging and Final Exam - The first three classes were used to generally describe tracing the flow of programs, inserting debug flag statements, etc. Quiz 4 was done during the third class. On Thursday the final was described and discussed. The final started on Friday with an in class portion consisting of five problems or procedures (simple math, science, accounting, etc.) to be programmed.

The following computerized lessons are available to you. Your school has a small minicomputer with five teletype terminals. There will be about 10 to 15 students in your class. The set of programs available contains three tutorials (linear programmed instruction), four drills, seven quizzes, and one fairly long test. You can assign any or all of these fifteen activities to each student individually, and may change the activities as often as you wish. The student can choose to do only those options you assign him, and do each only once. Following are the details of the options, including average expected execution times:

Tutorial 1: Rules for variable names, constants, and arithmetic expressions. 30 minutes.

Tutorial 2: Syntax and meaning of assignment and output statements, with statement numbering and arithmetic operators. 30 minutes.

Tutorial 3: Syntax and meaning of branching statements and the END statement. Assumes knowledge of statement numbering. Also teaches logical operators. 30 minutes.

Drill 1: Syntax of all statements. The student is shown statements and must choose one of five alternative answers. The four alternatives are the possible corrected forms, with one alternative saying that the original statement was correct. The students must determine which corrected form is in fact correct, or say that the original is correct. There are 50 items. Students spend as much time as desired, getting problems by random selection.

Drill 2: Producing statements. The student is given an English command and must choose the one of four BASIC commands which satisfies the English command, or say that none of them satisfy it. There are 50 items, same format as Drill 1.

Drill 3: Generating output. The student is given a short program using only the assignment, PRINT, and END commands. Each program has four alternative outputs. The student must choose the correct output for the program, or say that none is correct. There are 30 items, same format as Drill 1.

Drill 4: Generating output. The student is given a short program using all commands (those in Drill 3 plus the GOTO and IF-THEN) and again must choose the correct of four possible outputs, or say that none is correct. There are 30 items as in Drill 1.

Quiz 1: Generating output (same as Drill 4). The student must do 10 problems, 1 point each.

Quiz 2: Syntax. Each problem shows a combination of five variable names, constants, and arithmetic expressions. The student must identify the syntactically incorrect one. 10 problems, 1 point each.

Quiz 3: The student is given problems with arithmetic expressions containing only constants and must choose the correct numeric answer. 10 problems, 1 point each.

Quiz 4: Problems consist of five assignment and output statements and the student must identify the syntactically incorrect one. 10 problems, 1 point each.

Quiz 5: Each problem displays an arithmetic or logical operator. The student must choose the English phrase indicating its meaning. 20 problems, one half point each.

Quiz 6: Each problem displays an arithmetic expression. Alternatives are five English descriptions of the order of operations. Students must choose the correct one. 5 problems, 2 points each.

Quiz 7: Each problem consists of five sequence commands (GOTO's or IF-THEN's). The student must choose the syntactically incorrect one. 10 problems, 1 point each.

Test: 25 multiple choice exercises. 10 are of the Quiz 1 and Drill 4 variety. 15 consist of a program with five possible errors (including syntax, endless loops, no output, etc.) The student must indicate the error. Each question is worth 2 points on the final score. (Note: the previous teacher did NOT use this exam.)

Following is the subset of BASIC the students should learn.

Arithmetic operators

+ means 'positive' or 'add'
- means 'negative' or 'subtract'
* means 'multiply'
/ means 'divide'
**means 'to the power of . . .'
( ) specify order of operations

Order of operations

1) inner parentheses first
2) ** is done next
3) * and / are done next (from left to right)
4) + and - are done last (from left to right)

Logical operators

= means 'is equal to'
> means 'is greater than'
< means 'is less than'
>= means 'is greater than or equal to'
<= means 'is less than or equal to'
<> means 'is not equal to'

Constants (numbers)

A constant must have from 1 to 8 digits.
If the constant is positive it may optionally have a '+' to the left.
If negative, it MUST have a '-' to the left.
A decimal point is optional before the first, after the last, or between any digits. Only one decimal point is permitted in a single constant.

Variable names

A variable name may be 1) any letter, or 2) any letter followed by any digit 0 through 9.

Form of lines in BASIC

Each line consists of a 1 to 5 digit number (the line number), followed by a blank, followed by a statement. A line must be 80 columns or less.

Assignment statements

Assignment statements evaluate an arithmetic expression (containing constants, variables, and arithmetic operators) to a single constant, and put that result in a single variable.

Format: Variable=arithmetic expression
Rules: Blanks may be used freely between operators, constants, variables, and the equal sign.
Only a single variable may appear on the left of the '='.
Contiguous operators (other than parentheses) are not permitted except for + or - following *, /, or ** in which case + means 'positive' and - means 'negative'.
Contiguous variables and/or constants are not permitted.

Output statements

Format: PRINT (list of variables and literals separated by commas)
Rules: Each print statement starts a new output line. A print statement without a list will just skip a line.
Blanks may appear between variables, commas, literals, and the word PRINT.

A literal is a single quote followed by any string of symbols followed by a single quote. To put a single quote inside the literal, you must use two single quotes.

Variable names will be replaced by their numeric value at the time of the statement execution.
No spaces will come between members of the list on output unless put there using literals.
Decimal points will only be put in numbers if there are non-zero digits to the right of the decimal point.

Sequencing statements - GO TO

Format: GO TO (Line number) or GOTO (line number)
Rules: The line number must belong to a line appearing in the program. Any number of spaces may precede or follow the GO and the TO. This statement will cause execution to transfer to the statement with the line number indicated.

Sequencing statements - IF THEN

Format: IF (arith. exp.) (log. rel.) (arith. exp.) THEN (Line number)
Rules: Any number of spaces may precede the IF. At least one space is required after it.
At least one space precedes and follows the THEN.
The line number must correspond to one of the statements in the program. The arithmetic expression may be any permitted in an assignment.
Only one logical relation is permitted.
Usage: The two expressions are evaluated to single constants, and the constants compared to see if the logical relationship is true or false. If false, the next statement in the program is executed. If true, execution transfers to the statement with the line number indicated after the THEN.

END statement

The END statement stops execution of the program. It must be the last line in the program and have the largest line number.
Format: END
Rules: Any number of spaces are permitted before or after the END.

Character Set

The following characters are permitted:
letters: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
digits: 0 1 2 3 4 5 6 7 8 9
special: + - * / = ' ' . , ? ; : ( ) space < >

Execution order rules

Execution begins at the first statement and normally proceeds from each statement to the one immediately following it (i.e. having the next larger line number).
GOTO or IF THEN statements may branch to a line other than the next one, after which execution continues normally.
Execution ends when the END command is executed, or an error occurs, Errors included dividing by zero, overflow, and underflow.
Programs with syntax errors will not begin execution.

Tasks on Question Episodes

(Background for these tasks is found in ISEAC #4e.)

1. Write a BASIC program to present just a single question. Choose the question from your teaching subject. Be sure to include all the provisions required for an optionally designed computerized question episode.

Tasks on Lesson Evaluation

(Background for these tasks is found in ISEAC #5e.)

1. Select a computer lesson in your teaching field (and preferably, available on a computer system accessible to you). Conduct an evaluation of this lesson from the perspective of a teacher considering the implementation of the lesson. Prepare to discuss and to defend your evaluation of the lesson to colleagues. Also, prepare to describe any desired revisions to the lesson. If off-line documentation or materials are available for the lesson be sure to include these in your assessment.

Tasks on Tutorial CAI

(Background for these tasks is in ISEAC #6e.)

1. Review the description of a tutorial lesson (ISEAC #6e). On a computer system (or in a computer publication), locate a lesson described as a tutorial. Determine the extent to which the lesson meets the (optimal) description. Also, describe alterations you would make so that the lesson better fits the model of a tutorial.

Tasks on Drill and Practice

(Background for these tasks is found in ISEAC #7e.)

1. Prepare a list of five situations or topics in your teaching field for which mastery (and, hence, practice) is an appropriate goal. For each topic, describe the criterion such mastery should attain.

2. In ISEAC #7e there is a drill and practice utility program written in BASIC. Using this program, prepare a practice pool appropriate for achieving mastery in one of the contexts you identified in the above task. Be sure that the practice pool contains all elements necessary for effectively facilitating mastery in this context.

Tasks on Individualization and CMI

(Background for these tasks may be found in ISEAC #11e.)

1. The following list contains activities and decisions required of a classroom teacher. (They are arranged in no particular order.)

Should homework be assigned?
Are students present?
Record attendance.
Should today's lesson be evaluated?
Score tests over daily lessons.
Compute course grades.
Assign comprehensive evaluation task.
Decide course grading procedure.
Administer test on daily lesson.
Comprehensive evaluation by test or task?
Lesson evaluation by test or task?
Assign course grades.
Evaluate remedial work.
Administer lesson test.
Is the unit completed?
Report course grades.
Administer comprehensive test.
Evaluate comprehensive task against criteria.
Assign remedial work.
Lesson task performance satisfactory?
Observe performance on comprehensive task.
Assign lesson evaluation task.
Report student test grades.
Comprehensive evaluation needed.
Assign lesson test grades.
Report attendance to administrator.
Course completed?
Score comprehensive test.
Does test show learning satisfactory?
Present instructional lesson.
Assign comprehensive test grades.
Comprehensive performance satisfactory?
Report lesson test grades.
Collect lesson data on performance task.

Arrange these events into a course management flow chart. Add events where needed. Then determine those events which a computer system may implement, and the hardware or program facilities necessary for same.

2. For the course management flow chart in the previous task, select any event--decision event--branching event portion of the instructional flow. Using the CAISP system (see ISEAC #23e) or another CMI utility, implement specific materials from your teaching subject for this portion. Include as many additional individualizing features as possible.

3. Refer to the bibliography of ISEAC #11e. For each of the references by a) Baker, b) Grate, Rueve, and Moore, c) Scanlon and Connolly, and d) Allen:

i. Identify what the author defines as basic functions or characteristics of a CMI system.

ii. What distinction (if any) does each author make between CMI activities and instruction presentation activities? Provide examples wherever possible.

Tasks on Simulation

(Background for these tasks may be found in ISEAC #8e.)

1. Select any instructional simulation available to you. Identify in it:

a. Data gathering choices and consequences.
b. Management choices and consequences.
c. Consultation choices and consequences.
d. Valuing choices and consequences.

Also, flow chart an optional solution path for the simulation and at least one non-solution path.

2. A fellow teacher comes to you for a suggestion on using simulations in the teaching of their subject. With a colleague, role play this situation for a variety of teaching subjects,including your own. What simulations are available? How might they be used in a lesson? What teacher preparation would be necessary?

3. Many times during one's education there are opportunities to make decisions about courses to take. There are many effective and ineffective ways such may be done. Assume that you are building a simulation of this phenomenon.

a. Write an opening scene.
b. Identify the choice-consequence elements of an optimal solution path.
c. List a variety of other data gathering, managing, consulting, or valuing elements. Try to include a variety of qualities (from aborting to necessary) for each type.

4. Many times in teaching it is convenient to have both a batch operable and an interactive version of a particular simulation. Select any simulation appropriate for your teaching field. Make a companion version of it in the alternate execution mode.

Tasks on Instructional Games

(Background for these tasks may be found in ISEAC #9e.)

1. Select a popular game used in your teaching subject. Analyze the game and identify what you think to be the learning outcomes of playing the game. Be sure to consider higher level cognitive objectives as well as skill objectives. With a colleague, role play a conversation between a teacher using this game and a parent expressing concern over the level of enjoyment (rather than learning) in the class.

Tasks on Computerized Testing

(Background for these tasks may be found in ISEAC #12e.)

Several CATC systems exist. You may wish to study the Classroom Teacher Support System (CTSS) as implemented in Los Angeles City Unified School District or the Student Oriented Classroom Analysis and Test Evaluation System (SOCRATES) as implemented in the California State University and Colleges. In Illinois, at the secondary level, there is a CATC system developed by Paschke at Niles Township High Schools in Skokie. The system is called GENIE as in GENerator of Individualized Exams.

Having once studied a CATC system you may wish to design your own--thus your task follows:

1. On one sheet of paper, use a flow or block diagram to describe the broad functions desired in your CATC system.

2. On another sheet of paper, give examples of the type of information you imagine will exist in your data file(s) once your system is operational. Be sure to estimate the quantity of such information as well.

3. Some people identify five aspects of computer literacy as follows:

A) Hardware
B) Software
C) Applications
D) General knowledge of the industry
E) Social impact

Pick one of the above areas and design 10 questions (or an algorithm for generating 10 questions) for input to your CATC system. If your questions are multiple choice, include each alternative answer (be sure to identify which is the correct answer). Include values for any "assigned" attributes you are using in your system.

4. Here is data from a recent test you have given your class. Your CATC system gives you this data on the ten items.

Item Analysis

This table shows how the students responded to each question.
(How many answered A, B, C, Etc.)
For example: 0 Students chose 'A' for question 1.

NO.	A	B	C	D	E	Blank	Total Missed
1	0	1	0	*5	1	0	2
2	*5	0	0	1	1	0	2
3	0	0	2	2	*3	0	4
4	0	3	*3	1	0	0	4
5	0	*5	1	0	1	0	2
6	0	1	1	1	*4	0	3
7	*5	0	1	1	0	0	2
8	1	0	*3	3	0	0	4
9	*2	2	2	0	1	0	5
10	2	0	*5	0	0	0	2

Student Scores:

ID#	Score	%	Questions missed
026	7	70	1, 8, 9
025	8	80	1, 3
024	7	70	3, 4, 5
029	8	80	2, 9
027	4	40	2, 4, 6, 7, 8, 9
023	2	20	3, 4, 5, 6, 7, 8, 9, 10
022	4	40	3, 4, 6, 8, 9, 10

Here are the scores listed in decending order.

Rank	ID#	Score	%
1	025	8	80
2	029	8	80
3	026	7	70
4	024	7	70
5	027	4	40
6	022	4	40
7	023	2	20

Unfortunately, this system does not provide item difficulty, and discrimination indices.

You are to write an analysis of this test to include the following:

a. What action would you recommend for individual test items?

b. What action would you recommend for individual students?

c. What action would you recommend for the class as a whole?

Include all calculations you make in your report.

Tasks Related to Hardware Operation

(For the most part, background for these tasks is derived from using computers.)

1. Computing devices (terminals, storage devices, mainframes, etc.) usually have a variety of switches or other variable controls outside the box containing each of them. These control the functioning of the device. In school environments these indicators can get changed in a variety of unknown ways. Also, connections to other devices may become dissengaged. It then may be the teacher's task to get the device back in operating mode.

With a colleague, take turns troubleshooting miss-set switches/indicators on the outside of a variety of computing devices. To do this, one of you (in the absence of the other) first will record the correct settings of all external controls on the device and the state of connectors to other devices. Then this person arbitrarily will change one of the controls or disengage one of the connections. The other person then will be summoned to return the device to operating state. You should do this task a little competitively and try to stump each other. (When finished with the task, be sure to return all controls to their original state).

Tasks on Hardware Purchasing

1. Several persons should team for this activity. Select a particular school computing task (e.g. attendance, teaching programming, managing foreign language instruction, etc.). Identify computer system requirements for such a task (both hardware and software). Prepare these in the form of a request for dealer bids on the services. Wherever appropriate include tables for organizing data presented by a prospective dealer.

2. If possible in nearby schools, obtain a copy of a vendor's proposal for computer services. Be sure, also, to obtain as much detail as is available about the preceding needs study and written request for bids. Analyze each of these documents and present an evaluation of each of them, together with your assessment of the probable success in obtaining satisfactory computer services.